MULTIMEDIA UNIVERSITY

STUDENT ID NO

□□□□□□□□□□

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2016/2017

ECP1026 – ALGORITHM AND DATA STRUCTURE

(All sections / Groups)

10 MARCH 2017
9:00 a.m – 11:00 a.m
(2 Hours)

INSTRUCTION TO STUDENTS

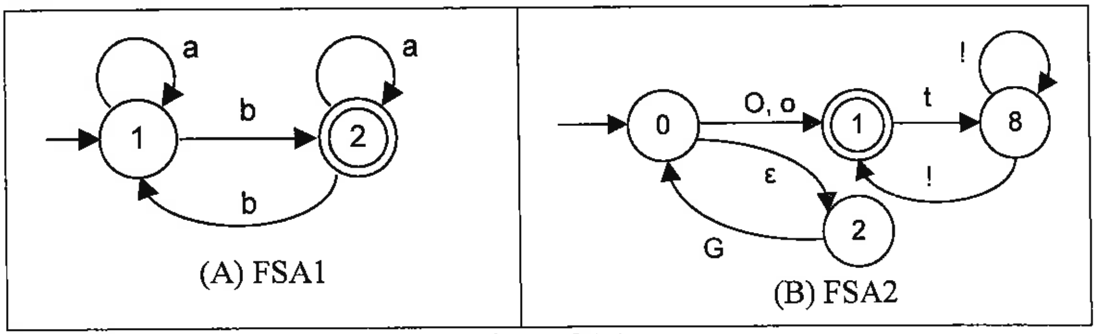
1. This Question paper consists of 5 pages including cover page with 4 Questions only.

2. Attempt **ALL** questions. All questions carry equal marks and the distribution of marks for each question is given.

3. Please write all your answers in the Answer Booklet provided.

Question 1

(a) Consider the finite state automata (FSA) given in the Figure Q1.1.



(A) FSA1

(B) FSA2

Figure Q1.1

i) Determine if each of the FSA is a deterministic or non-deterministic FSA. Give one reason to support your choice. [2+2 marks]

ii) Express each of the above FSA with a regular expression (RE). [4+4 marks]

(b) Multiplication in binary is much simpler than in decimal. This is because the product by 0 or 1 is just 0 or the same number, respectively. Therefore, the multiplication of two binary numbers comes down to calculating the partial products (which will be either zeros or a shifted copy of the multiplicand), and adding them together.

Multiplication of a binary number by two can be performed effectively by adding a zero at the multiplicand. For example,

		1	**1**	**0**	(this is 6 in decimal)
×			1	0	(this is 2 in decimal)
		0	0	0	(this is 110×0)
+	1	1	0		(this is 110×1, shifted one position to the left)
	1	**1**	**0**	0	(this is 12 in decimal)

i) A Turing machine (TM) that multiplies a given binary number by two is to be designed. Represent the Finite State Machine (FSM) controller in the state diagram form. You may assume the read/write head of the FSM controller begins at the left-most non-blank character on the input tape. [8 marks]

ii) Run the TM on an input tape containing a binary string of 101. During each step of the process, show clearly the current state, the contents of the tape and the character that is being read by the read/write head. [5 marks]

Continued...

Question 2

(a) A complex number is expressed in the form $a + bi$, where a and b are real numbers, and i is the imaginary unit. In this expression, a represents the real part and b the imaginary part of the complex number.

Addition of two complex numbers is performed by separately adding the real parts, and the imaginary parts, i.e.,

$$(m + ni) + (p + qi) = (m + p) + (n + q)i$$

Figure Q2.1 shows the skeleton of a C program consisting of a user-defined function called `addCom` that calculates the addition of two complex numbers, and places the results in its pointer argument. Based on the comments given, complete the program.

```
#include <stdio.h>
//i) Define suitable record type for a complex number

//ii) Define function addCom

int main(){
  //iii) Declare record variables c1, c2 and sum

  //iv) Prompt user and input for c1 and c2

  //v) Call addCom; Passing c1 and c2 by value, and
  //        sum by reference to addCom

  //vi) Print summation result
  return 0;
}
```

Figure Q2.1

[16 marks]

(b) Consider the weighted graph given in Figure Q2.2.

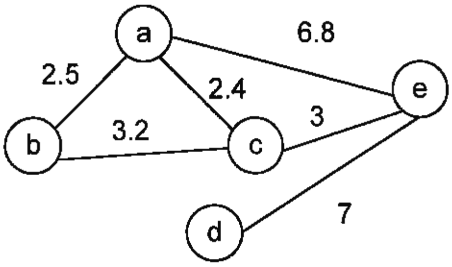


Figure Q2.2

i) Represent the graph in adjacency matrix form. [3 marks]

ii) Write C declaration code that creates an array implementation of this graph. Use -1 to indicate no direct path from one node to another. [6 marks]

Continued...

Question 3

(a) Powers of a number, i.e., $base^{exponent}$ where *exponent* is positive, can be calculated by the following mathematical expression,

$$base^{exponent} = \begin{cases} 1 & \text{if } exponent = 0, \\ base \times base^{exponent-1} & \text{otherwise.} \end{cases}$$

Write a recursive function `power` that computes and returns the value of *base* raised to power of *exponent*, where *base* and *exponent* are function arguments. The function prototype is given by: `int power(int base, int exp);`

[6 marks]

(b) Specify TWO disadvantages of using recursive implementation. [3 marks]

(c) Consider the function definitions `sum1` and `sum2` that calculates the summation of all integer values in the range of 1 and *n*-1, as given in Figure Q3.1.

```
int sum1(int n) {
  int sum;

  sum = n*(n-1)/2;

  return sum;
}

int sum2(int n) {
  int i, sum=0;

  for(i=1; i<n; i++)
    sum = sum + i;

  return sum;
}
```

Figure Q3.1

i) State clearly the time complexity for each statement (if any) of functions `sum1` and `sum2`. Calculate the worst-case time complexity of both functions. [6+2+2 marks]

ii) Estimate the Big-O of both functions. [2+2 marks]

iii) Based on your finding in part (ii), comment on the efficiency of both implementations in term of execution time. [2 marks]

Continued...

Question 4

(a) State TWO conditions under which binary search of a list of items is preferable over sequential search. [2 marks]

(b) Consider the binary search algorithm performed on a sorted array in Figure Q4.1.

```
while low index is smaller or equal to high index
   locate the middle element

   if the middle element matches with the search key
      return the middle index
   otherwise, if the search key is smaller than the middle element
      adjust high index so that first-half of array is searched in the next cycle
   otherwise
      adjust low index so that second-half of array is searched in the next cycle

return not_found
```

Figure Q4.1

Write a C function that implements binary search using iteration. The function prototype takes the form of:

```
int binSearch(int a[], int low, int high, int key);
```

[13 marks]

(c) Given the following items stored in an integer array:

2, 5, 8, 10, 25, 14, 32, 40, 28, 46, 22, 50

i) Suggest a suitable sorting method to arrange these items in increasing order. Justify your choice. [4 marks]

ii) By hand, trace through the steps when implementing the sorting method suggested in part (i) on these items. Underline the item that is being processed at each step. [6 marks]

End of Paper